Mapping Cores onto Network-on-Chip

ABSTRACT

The paper addresses the problem of topological mapping of intellectual properties (IPs) on the tiles of a mesh-based network on chip (NoC) architecture. The aim is to obtain the Pareto mappings that maximize performance and minimize the amount of power consumption. As the problem is an NP-hard one, we propose a heuristic technique based on evolutionary computing to obtain an optimal approximation of the Pareto-optimal front in an efficient and accurate way. At the same time, two of the most widely-known approaches to mapping in mesh-based NoC architectures are extended in order to explore the mapping space in a multi-criteria mode. The approaches are then evaluated and compared, in terms of both accuracy and efficiency, on a platform based on an event-driven trace-based simulator which makes it possible to take account of important dynamic effects that have a great impact on mapping. The evaluation performed on real applications (an MPEG-4 codec) confirms the efficiency, accuracy and scalability of the proposed approach.

1. INTRODUCTION

The continuous reduction in the time-to-market required by the telecommunications, multimedia and consumer electronics market makes full-custom design of an interconnection system inappropriate and has led to the definition of design methodologies focusing on design reuse. This is confirmed by the great standardization effort made by the VSI Alliance [20] and the development, by the major EDA and Semiconductor companies, of on-chip interconnection systems that are easy to integrate and scale [9, 3]. Although, however, they are good solutions for current SoCs integrating fewer than 5 processors and rarely more than 10 master buses, their use in next-generation systems, which are likely to integrate hundreds of elements, seems hardly feasible. The limiting factor is mainly the topological organization of the interconnection between the various units, which will substantially remain bus-based. As regards performance, the continuous reduction in gate delays and increase in wiring delays will cause significant synchronization problems. In 50 nm technology, the projected chip die edge will be around 22 mm, with a clock frequency of 10 GHz. An optimistic estimate of the propagation delay for a signal crossing a chip diagonally (taking wire sizing and buffering into account) ranges between 6 and 10 clock cycles [18].

A type of architecture which lays emphasis on modularity and is intrinsically oriented towards supporting such heterogeneous implementations is represented by Network-on-Chip (NoC) architectures [7]. These architectures loosen the bottleneck due to delays in signal propagation in deep-submicron technologies and provide a natural solution to the problem of core reuse by standardizing on-chip communications. In this paper we will focus on mesh-based NoC architectures, in which resources communicate with each other via a mesh of switches that route and buffer messages. A resource is generally any core: a processor, a memory, an FPGA, a specific hardware block or any other IP compatible with the NoC interface specifications.

One of the most onerous tasks in this context is the topological mapping of the resources on the mesh in such a way as to optimize certain performance indexes (e.g. power, performance). Mapping is, in fact, a problem of quadratic assignment that is known to be NP-hard. It is therefore of strategic importance to define methods to search for a mapping that will optimize the desired performance indexes.

In this paper we present a multi-objective exploration approach for the mapping space of a mesh-based NoC architecture. The approach, based on evolutionary computing techniques, is an efficient and accurate way to obtain the Pareto mappings that optimize performance and power consumption. In addition, two of the most widely known approaches to topological mapping of IPs in a mesh-based NoC architecture [8, 15] have been extended to achieve multi-criteria optimization and have been compared with the approach proposed here.

2. PREVIOUS WORK

The problem of mapping in mesh-based NoC architectures has been addressed in three previous papers. Hu and Marculescu [8] present a branch and bound algorithm for mapping IPs/cores in a mesh-based NoC architecture that minimizes the total amount of power consumed in communications with the constraint of performance handled via bandwidth reservation. Murali and De Micheli [15] address the problem under the bandwidth constraint with the aim of minimizing communication delay by exploiting the possibility of splitting traffic among various paths. Lei and Kumar [13] present an approach that uses genetic algorithms to map an application, described as a parameterized task graph, on a mesh-based NoC architecture. The algorithm finds a mapping of the vertices of the task graph on the available cores so as to minimize the execution time.

These papers do not, however, solve certain important issues. The first relates to the mapping evaluation model used, which can be defined as “static”. The exploration algorithm decides which mapping to explore without taking important dynamic effects of the system into consideration. For example, failure to model the effects of bus contention causes components which communicate with each other more frequently to be clustered, whereas it may be more effective to separate components whose traffic flows overlap in time so as to increase the degree of concurrency. In the above-mentioned works, in fact, the application to be mapped is described using task graphs, as in [13], or simple variations such as the core graph in [15] or the application characterization graph (APCG) in [8]. These formalisms do not, however, capture important dynamics of communication traffic. They hypothesize worst-case conditions, which leads to several mappings being discarded and thus a highly conservative exploration. The second problem relates to the optimization method used. It refers in all cases to a single performance index (power in [8], performance in [15, 13]). As we will see in the section devoted to experiments, optimization of one performance index may lead to unacceptable values for another performance index (e.g. high performance levels but unacceptable power consumption). We therefore think that the problem of mapping can be more usefully solved in a multi-objective environment, i.e. one in which there is no single solution but a set of mapping alternatives (which we will indicate as Pareto mapping), each featuring a different tradeoff between performance indexes, from which the designer (or decision maker) will choose the most suitable.

The contribution we intend to make in this paper is to propose a multi-objective approach to solving the problem of mapping IPs/cores in mesh-based NoC architectures. The approach will use evolutionary computing techniques to explore the mapping space with the goal to optimize performace and power consumption. The mappings visited during the exploration process will be evaluated using a trace-based approach which gives an excellent combination of accuracy and efficiency features.

3. MULTI-OBJECTIVE EXPLORATION OF THE MAPPING SPACE

The mapping problem is an instance of a constrained quadratic assignment problem which is known to be *NP-hard*. The search for an optimal mapping (henceforward referred to as exploration) is also complicated when the concept of optimality is not limited to a single performance index (or objective) but comprises several contrasting indexes. The traditional approach to a multi-objective optimization is to aggregate the objectives into a single one by means of a *weighting mean*. The main drawback to this approach is that it does not cover the non-convex regions of the Pareto-front and requires several instances of the optimization algorithm to be run with different weights. In this section we present: 1) an approach to multi-objective mapping space exploration that uses evolutionary algorithms as the optimization strategy; 2) multi-objective extension of an exploration algorithm based on the branch-and-bound proposed in [8]; and 3) multi-objective extension of a variation of the exploration algorithm proposed in [15].

3.1 Problem Formulation

If C is the set of cores, and T the set of tiles, we will use the term *mapping* to indicate an injective and surjective function $M : C \rightarrow T$ that associates the tile $t \in T$ on which c is mapped with each $c \in C$.

Evaluating a mapping means obtaining the related performance indexes for a specific traffic scenario. If S indicates a traffic scenario, we define the *evaluation function*

$$\mathbf{V}(S,M) = (V_1(S,M), V_2(S,M), \ldots, V_n(S,M))$$

which yields the values of the n performance indexes relating to the mapping M for the traffic scenario S. In our case study, for example, the evaluation function corresponds to the simulation framework (described in [4]) and the performance indexes are those the platform is capable of measuring (power, communication latency, bandwidth, throughput, etc.). Evaluation of an incomplete mapping made up of a set of cores $C' \subset C$ with a traffic scenario S is performed by evaluating the mapping on a traffic S' obtained by filtering out all communication flows in which the source or destination is a core $c \in C'$.

Given a traffic scenario S and two mappings M_1 and M_2, M_1 can be said to *dominate* M_2 (which will be indicated as $M_1 \prec M_2$) if $V_i(S,M_1) \leq V_i(S,M_2),\ i \in \{1,2,\ldots,n\}$ and there exists at least one $j \in \{1,2,\ldots,n\}$ such that $V_j(S,M_1) < V_j(S,M_2)$. The *set of Pareto mappings* is a set of mappings that do not dominate each other. The Pareto front is the image of the evaluation function for the set of Pareto mappings. If $\mathscr{M}$ is the set of all possible mappings, the Pareto-optimal set $\mathscr{P}$ is the set of Pareto mappings such that $\nexists\, M \in \mathscr{M} : M \prec M',\ M' \in \mathscr{P}$.

The aim of the approach we propose is to obtain as accurate an approximation as possible of the Pareto-optimal front by evaluating (visiting) as few mappings as possible.

3.2 GA-based Multi-Objective Exploration of the Mapping Space

The use of evolutionary algorithms (EAs) as a multi-objective optimization technique is of increasing appeal. The fields of application are numerous, including among others computer science, engineering, economics, finance, industry, physics, chemistry, and ecology. EAs have been demonstrated to be very powerful and generally applicable for solving difficult multi-objective problems. Such algorithms create an interesting alternative to other approaches since they can be scaled with the problem size and can be easily run on parallel computer systems. In VLSI design, EAs have been applied to a very broad range of problems: in problems relating to layout such as partitioning [1], placement and routing [19], in design problems including power low-power synthesis [6], technology mapping [11] and netlist partitioning [2].

In this paper we propose the use of a heuristic technique based on EAs for multi-objective mapping space exploration. More precisely, we use the Strength Pareto Evolutionary Algorithm (SPEA) [21] which maintains an external set to preserve the nondominated solutions encountered so far besides the original population. The chromosome is a representation of the solution to the problem, which in this case is described by the mapping. Each tile in the mesh has an associated gene which encodes the identifier of the core mapped in the tile. In an $n \times m$ mesh, for example, the chromosome is formed by $n \times m$ genes. The i-th gene encodes the identifier of the core in the tile in row $\lceil i/n \rceil$ and column $i\%m$ (where the symbol % indicates the modulus operator).

The *crossover* and *mutation* genetic operators were have been suitably redefined. More specifically, a crossover between two mappings M_f and M_m generates a new mapping M_s constructed as follows. The dominant mapping between M_f and M_m is chosen. Its hot-spot core is remapped on a tile in a random position in the mesh, thus providing the new mapping M_s. Figure 1 describes the crossover operator. Where the function `Swap(`M`,`i`,`j`)` exchanges the i-th tile with the j-th tile from mapping M.

The mutation operator acts on a single mapping M to obtain the mutated mapping M' as follows. A tile T_s from mapping M is chosen at random. Indicating the core in the tile T_s as c_s and c_t as the core with which c_s communicates most frequently, c_s is remapped on a tile adjacent to T_s so as to reduce the distance between c_s and

```
Mapping XOver(Mapping $M_f$ , Mapping $M_m$)
{
    Mapping $M_s$;

    if ($M_f \prec M_m$)
        $M_s$ = $M_f$;
    else
        $M_s$ = $M_m$;

    Swap($M_s$ , HotSpot($M_s$) , Random($\{1, 2, \ldots, N^2\}$));

    return $M_s$;
}
```

Figure 1: Crossover operator.

c_t by a hop, thus obtaining the mutated mapping M'. Figure 2 describes the mutation operator. The RandomTile(M) function

```
Mapping Mutate(Mapping $M$)
{
    Mapping $M'$ = $M$;

    Tile $T_s$ = RandomTile($M'$);
    Core $c_s$ = $M'^{-1}(T_s)$;
    Core $c_t$ = MaxCommunication($c_s$);
    Tile $T_t$ = $M'(c_t)$;

    Tile $T_s'$;
    if (Row($M'$ , $T_s$) < Row($M'$ , $T_t$))
        $T_s'$ = North($M'$ , $T_s$);
    else if (Row($M'$ , $T_s$) > Row($M'$ , $T_t$))
        $T_s'$ = South($M'$ , $T_s$);
    else if (Col($M'$ , $T_s$) < Col($M'$ , $T_t$))
        $T_s'$ = East($M'$ , $T_s$);
    else
        $T_s'$ = West($M'$ , $T_s$);

    Swap($M'$ , $T_s$ , $T_s'$);

    return $M'$;
}
```

Figure 2: Mutation operator.

gives a tile chosen at random from mapping M. The MaxCommunication(c) function gives the core with which c communicates most frequently. The Row(M,T) and Col(M,T) functions respectively give the row and column of the tile T in mapping M. Finally, the North, South, East, West(M,T) functions give the tile to the north, south, east and west of the tile T in mapping M.

The definition of suitable and more effective genetic operators has a great impact on the results of the optimization. This is not, however the aim of this paper and remains a topic for future research.

3.3 Pareto-based Branch-and-Bound Approach

In [8] Hu and Marculescu present an approach using branch-and-bound as the mapping space exploration strategy. The approach is, however, a mono-objective one. In this subsection we will extend their approach in order to perform multi-objective exploration of the mapping space. We will call our approach *Pareto-based Branch-and-Bound* (*PBBB*).

Let $\{c_1, c_2, \ldots, c_{N^2}\}$ be the set of cores in the system in decreasing order with respect to the communication traffic. The core c_1 can be mapped on any of the N^2 tiles in the mesh. These N^2 mappings generate the first layer of a tree which is the starting point for the branch-and-bound algorithm. For each first-level mapping the core c_2 can be mapped on any of the $N^2 - 1$ free tiles, thus generating a second level $N^2 \times (N^2 - 1)$ mappings. This is the *branch* phase of the algorithm and is described in pseudo-code in Figure 3. Where

```
Mappings Branch(Mappings $\mathscr{M}$ , Core $c$)
{
    Mappings $\mathscr{M}'$ = $\emptyset$;

    for ($M \in \mathscr{M}$)
        $\mathscr{M}'$ = $\mathscr{M}'$ $\bigcup$ MakeMappings($M$ , $c$);

    return $\mathscr{M}'$;
}
```

Figure 3: Branch phase of the branch-and-bound algorithm.

the MakeMappings(M,c) function, given a mapping template M and a core c, yields a set of mappings obtained by mapping c on each free tile in M.

Each mapping at this level is evaluated (simulated) and then characterized according to the optimization objectives, which in our case are power and delay. The dominated mappings are discarded, while the branch and bound phases are reiterated on the survivors. This is the *bound* phase of the algorithm as described in pseudo-code in Figure 4. Where the ExtractPareto($\mathscr{M}$)

```
Mappings Bound(Mappings $\mathscr{M}$)
{
    Mappings $\mathscr{M}'$ = ExtractPareto($\mathscr{M}$);

    if ($|\mathscr{M}'| > T_{pbbb}$)
        Pruning($\mathscr{M}'$ , $T_{pbbb}$);

    return $\mathscr{M}'$;
}
```

Figure 4: Bound phase of the branch-and-bound algorithm.

function extracts the non-dominated mappings from the set $\mathscr{M}$. To prevent the algorithm from degenerating the bound phase is followed by a further pruning phase. Let us indicate the set of mappings generated by the bound phase as $\mathscr{M}$. If $|\mathscr{M}| > T$ (where T is a user-defined threshold) $|\mathscr{M}| - T$ mappings are eliminated at random from $\mathscr{M}$. The Pruning($\mathscr{M}, T_{pbbb}$) function randomly eliminates mappings from a set $\mathscr{M}$ if the cardinality of this set exceeds a threshold T_{pbbb} in such a way as to make the cardinality of $\mathscr{M}$ equal to T_{pbbb}.

The branch and bound phases are reiterated until all the cores have been mapped. For example, indicating the mappings obtained in the bound phase as $M_1, M_2, \ldots, M_n$, the core c_3 will be mapped for each of them on to the $N^2 - 2$ possible tiles. The $n \times N^2 - 2$ mappings will be the third level of the tree. The algorithm terminates when all the cores have been mapped and the leaves of the tree will be the Pareto mappings. A pseudo-code description of *PBBB* is given in Figure 5. Where the SortByTraffic(C) function orders the set of cores C according to the communication traffic.

3.4 Pareto-based NMAP Approach

Murali and De Micheli in [15] propose NMAP, an algorithm that maps the cores in a mesh NoC architecture with the aim of minimizing the average communication delay. In this subsection we will extend NMAP to perform a multi-objective exploration of the mapping space. Unlike [15], however, we will refer to a routing XY. We will call this approach *Pareto-based NMAP* (*PBNMAP*).

```
Mappings PBBB(Cores $C$)
{
    Cores $C_s$ = SortByTraffic($C$);
    Mappings $\mathscr{M}$ = MakeMappings($\emptyset$ , $C_{s,1}$);
    for ($c \in C_s \setminus \{C_{s,1}\}$) {
        $\mathscr{M}$ = Branch($\mathscr{M}$ , $c$);
        $\mathscr{M}$ = Bound($\mathscr{M}$);
    }

    return $\mathscr{M}$;
}
```

Figure 5: Pareto-based branch-and-bound approach.

The algorithm comprises two phases. In the first the cores featuring the largest amount of communication traffic are mapped onto the central tiles in the mesh (i.e. the $(N-2)\times(N-2)$ tiles with the greatest numbers of neighbours). The remaining cores are then ordered in decreasing order with respect to the communication traffic they have with the cores mapped in the previous phase. The first, c_1, is mapped onto each of the $4\times(N-1)$ remaining tiles. The $4\times(N-1)$ are evaluated and those that are dominated are discarded. If $\mathscr{M}_1$ is the set of non-dominated mappings, the algorithm is reiterated for each $M \in \mathscr{M}_1$ with c_2 and so on until the last core $c_{4(N-1)}$ has been mapped and the set of Pareto mappings $\mathscr{M} = \mathscr{M}_{4(N-1)}$ has been obtained. Figure 6 give the pseudo-code for this first phase. Where the Map(M,c,row,col) function maps

```
Mappings PBNMAP_1st(Cores $C$)
{
    Cores $C_s$ = SortByTraffic($C$)
    Mapping $M$;
    for ($i \in \{1,2,\dots,(N-2)\times(N-2)\}$)
        Map($M$ , $C_{s,i}$ , $(i-1)/(N-2)+1$ , $(i-1)\%(N-2)+1$);

    Cores $C2_s$ =
        SortByC2CTraffic($\{C_{s,(N-2)*(N-2)+1},\dots,C_{s,N^2}\}$ ,
                         $\{C_{s,1},\dots,C_{s,(N-2)*(N-2)}\}$);
    Mappings $\mathscr{M} = \emptyset$;
    for ($c \in C2s$) {
        $\mathscr{M}$ = MakeMappings($\mathscr{M}$ , $c$);
        $\mathscr{M}$ = ExtractPareto($\mathscr{M}$);
    }

    return $\mathscr{M}$;
}
```

Figure 6: First phase of the Pareto-based NMAP approach.

core c onto the tile in row *row* and column *col* of the mapping M. The SortByC2CTraffic(C_a,C_b) function sorts the cores in the set C_a according to the communication traffic they have with the cores in the set C_b.

In the second phase, the mapping of cores c_i and c_j is inverted for each mapping $M \in \mathscr{M}$ and each pair (c_i, c_j), thus obtaining the new mapping M'. The algorithm proceeds with the next pair on the mapping M or M' according to whether M dominates M' or M' dominates M. If M and M' are Pareto mappings, the algorithm proceeds with the next pair on both mappings. A pseudo-code description of this phase is given in Figure 7, while Figure 8 describes the main program.

```
Mappings PBNMAP_2nd(Mappings $\mathscr{M}$ , Cores $C$)
{
    for ($i \in \{1,2,\dots,N^2-1\}$)
        for ($j \in \{i+1,i+2,\dots,N^2\}$) {
            Mappings $\mathscr{M}_n = \emptyset$;
            for ($M \in \mathscr{M}$) {
                Mapping $M'$ = Swap($M$ , $i$ , $j$);
                if ($M' \prec M$)
                    $\mathscr{M}_n$ = $\mathscr{M}_n \bigcup \{M'\}$;
                else if ($M \prec M'$)
                    $\mathscr{M}_n$ = $\mathscr{M}_n \bigcup \{M\}$;
                else
                    $\mathscr{M}_n$ = $\mathscr{M}_n \bigcup \{M,M'\}$;
            }
            $\mathscr{M}$ = ExtraxtPareto($\mathscr{M}_n$);
    }

    return $\mathscr{M}$;
}
```

Figure 7: Second phase of the Pareto-based NMAP approach.

```
Mappings PBNMAP(Cores $C$)
{
    Mappings $\mathscr{M}$;

    $\mathscr{M}$ = PBNMAP_1st($C$);
    $\mathscr{M}$ = PBNMAP_2nd($\mathscr{M}$ , $C$);

    return $\mathscr{M}$;
}
```

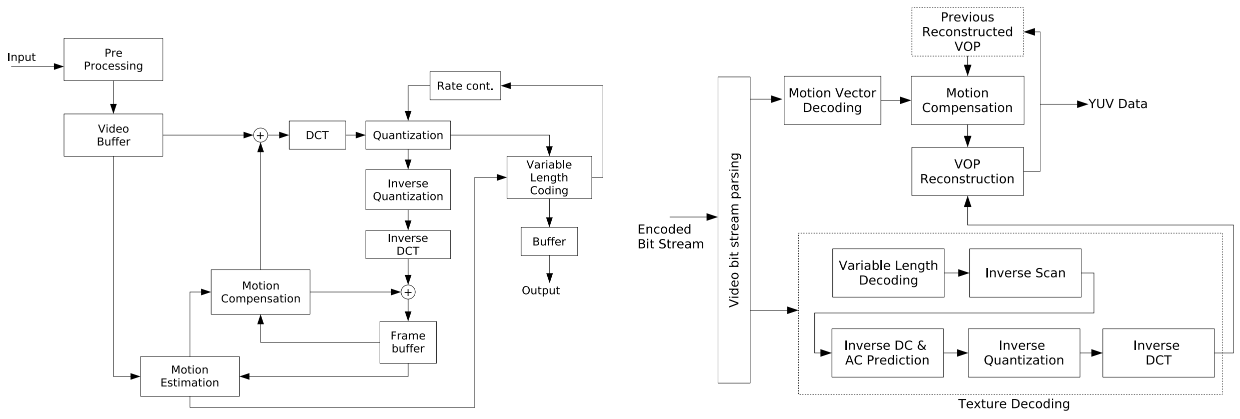
Figure 8: Pareto-based NMAP approach.

4. EXPERIMENTS

In order to evaluate the various approaches in real traffic scenarios, an MPEG-4 simple profile @ level 2 codec was used as a case study [17]. A general block diagram of the encoder and decoder is shown in Figure 9.

For the hardware/software partitioning reference was made to the MoVa architecture described in [10]. It adopts a macroblock-based pipeline with 4 stages for the encoder and 3 for the decoder. More specifically, the encoding section performs coarse motion estimation in the first stage, fine motion estimation fine and motion compensation in the second stage, discrete cosine transform and quantization in the third stage, and finally reconstruction and production of the stream in the fourth stage. In the decoding section, the first stage involves variable length decoding of each data stream; in the second stage it performs sequential inverse cosine transformation, inverse quantization and motion compensation; the third and final stage is reconstruction.

To obtain the traffic traces the C application implementing the codec [12] was modified with the addition of a monitor code to record the volume of incoming and outgoing traffic in the various functional blocks into which the application is partitioned. Table 1 shows the 16 cores implementing the codec. They were characterized in terms of timing by using the clock cycle data in [10] for the execution of each operation (DCT, MC, etc.). For power characterization, we used the mean values given in the datasheets [14, 16]. For the interconnection system we used an approach similar to the one presented in [8]. To characterize the switches, a 5x5 switch was implemented in VHDL following the architecture described in [5]. It was synthesized with a Synopsys Design Compiler using the Virtual Silicon $0.13\mu m$, $1.2V$ technological library and analyzed using Synopsys Design Power using different random input data streams for the inputs of the switch. The amount of power consumed by a flit for a hop switch was estimated as being $0.181nJ$. We assumed the tile size to be $2mm \times 2mm$ and that the tiles were arranged in a regular fashion on the floorplan. The load wire capacitance was set



(a) (b)

Figure 9: Block diagram of the MPEG-4 codec. (a) Encoder. (b) Decoder.

Core	*Description*
MEC	Motion estimation coarse
MEF	Motion estimation fine
MC	Motion compensation
VLC	Variable length coding
VLD	Variable length decoding
REC	Reconstruction
SP	Stream producer
DB	Deblocking
DCTQ	Discrete cosine transform & quantization
IQIDCT	Inverse discrete cosine transform & inverse quantization
RISC	32 bit risc microprocessor
VIM	Video input module
VOM	Video output module
ISC	Input stream controller
MEME	Encoder memory
MEMD	Decoder memory

Table 1: Cores implementing the codec.

to $0.50fF$ per micron, so considering an average of 25% switching activity the amount of power consumed by a flit for a hop interconnect is $0.384nJ$.

Figure 10 gives the power values and traffic clearing times for 10,000 random mappings. It also shows the Pareto fronts obtained by *GAMAP*, *PBNMAP*, and *PBBB*. As can be seen, the solutions obtained by *GAMAP* dominate those obtained by the other approaches. The figure also shows the good trade-off between delay and power (respectively equal to a factor of 3 for delay and 2.5 for power).

Figure 11(a) gives the number of simulations (i.e. mappings evaluated by *GAMAP*) for varying numbers of generations. It gives the number of simulations actually performed and those virtually performed if no caching mechanism had been used. Figure 11(b) gives the normalized delay and energy values for varying numbers of generations. As can be seen, in both cases no mappings that determine appreciable improvements in delay and energy consumption are found after the 20th generation. At the 20th generation *GAMAP* had only performed 840 simulations as compared with 2,670 by *PBNMAP* and 7,238 by *PBBB*, thus providing an exploration time speed-up of 3.2 and 8.6 respectively.

Finally, Figure 12 shows a point in the Pareto set obtained by *GAMAP*. The cores specific to the encoding section are shown against a dark gray background, whereas those specific to the decoding are against a white background. The cores shared by the encoder and

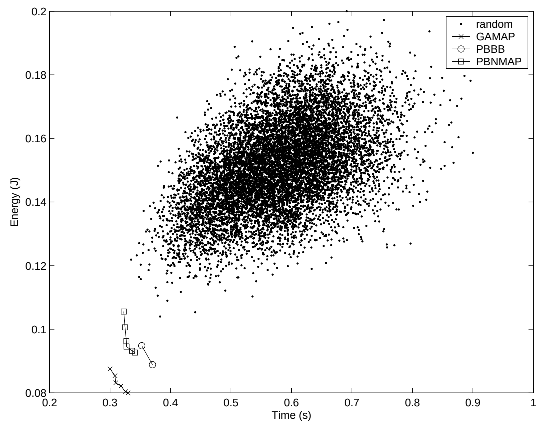
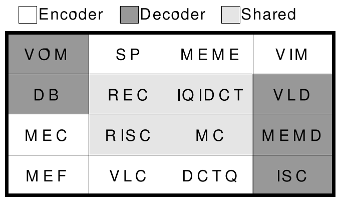


Figure 10: Evaluation of 10,000 random mappings and Pareto fronts obtained by *GAMAP*, *PBNMAP*, and *PBBB* for a 4x4 NoC in which the MPEG-4 codec is mapped on.


Encoder
Decoder
Shared

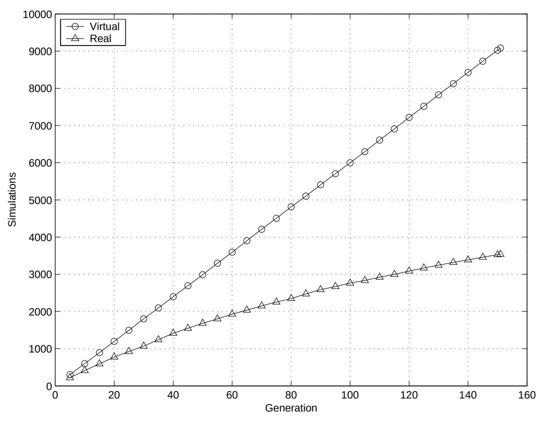
VOM	SP	MEME	VIM
DB	REC	IQIDCT	VLD
MEC	RISC	MC	MEMD
MEF	VLC	DCTQ	ISC

Figure 12: Pareto mapping for the l'MPEG-4 codec.

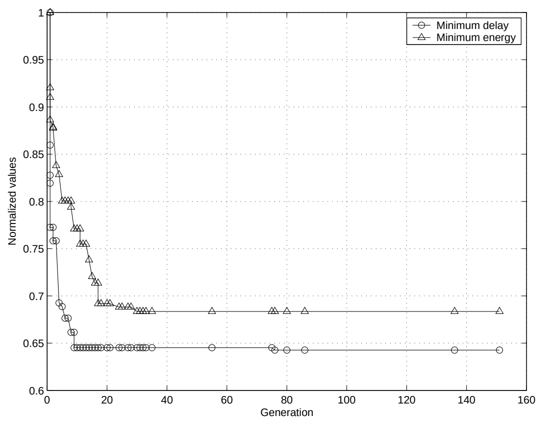
decoder are shown against a light gray background and have been mapped (in this case) in the centre of the NoC. In the decoding section, the cores VOM and DB are topologically separated from VLD, MEMD and ISC as there is no direct communication flow between these sets: they communicate by means of a ring represented by the core REC. In the encoding section there are also two separate parts which do not communicate directly but through the set of shared cores.

5. CONCLUSIONS

In this paper we have proposed a strategy for topological mapping of IPs/cores in a mesh-based NoC architecture. The approach uses heuristics based on multi-objective genetic algorithms to explore the mapping space and find the Pareto mappings that optimize performance and power consumption. At the same time, two



(a)



(b)

Figure 11: Number of (virtual and real) mappings evaluated by *GAMAP* in varying numbers of generations (a). Normalized minimum delay and power consumption values obtained by the *GAMAP* in varying numbers of generations (b).

of the most widely-known approaches to mapping in mesh-based NoC architectures have been extended in order to explore the mapping space in a multi-criteria mode. The approaches have been then evaluated and compared, in terms of both accuracy and efficiency, on a platform based on an un event-driven trace-based simulator which makes it possible to take account of important dynamic effects that have a great impact on mapping. The experiments carried out on a real application (an MPEG-4 encoder/decoder system) confirm the efficiency, accuracy and scalability of the proposed approach. Future developments will mainly address the definition of more efficient genetic operators to improve the precision and convergence speed of the algorithm. Evaluation will also be made of the possibility of optimizing mapping by acting on other architectural parameters such as routing strategies, switch buffer sizes, etc.

6. REFERENCES

[1] C. J. Alpert, L. W. Hagen, and A. B. Kahng. A hybrid multilevel/genetic approach for circuit partitioning. In *Fifth ACM/SIGDA Physical Design Workshop*, pages 100–105, Apr. 1996.

[2] C. J. Alpert and A. B. Kahng. Recent developments in netlist partitioning: A survey. *VLSI Journal*, 19(1–2):1–81, 1995.

[3] ARM. AMBA specification. `http://www.arm.com/`, May 1999.

[4] G. Ascia, V. Catania, and M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In *Second IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 182–187, Stockholm, Sweden, Sept. 8–10 2004.

[5] N. Banerjee, P. Vellanki, and K. S. Chatha. A power and performance model for network-on-chip architectures. In *Design, Automation and Test in Europe*, pages 1250–1255, Feb. 16–20 2004.

[6] M. S. Bright and T. Arslan. Synthesis of low-power DSP systems using a genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 5(1):27–40, 2001.

[7] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, Las Vegas, Nevada, USA, 2001.

[8] J. Hu and R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Asia & South Pacific Design Automation Conference*, Jan. 2003.

[9] IBM Corporation. The CoreConnect bus architecture. `http://www.ibm.com/`.

[10] S.-M. Kim, J.-H. Park, S.-M. Park, B.-T. Koo, K.-S. Shin, K.-B. Suh, I.-K. Kim, N.-W. Eum, , and K.-S. Kim. Hardware-software implementation of MPEG-4 video codec. *ETRI Journal*, 25(6):489–502, Dec. 2003.

[11] V. Kommu and I. Pomenraz. GAFAP: Genetic algorithm for FPGA technology mapping. In *European Design Automation Conference*, pages 300–305, 1993.

[12] C. Lampert, M. Militzer, and P. Ross. XviD MPEG4 core library. `http://www.xvid.org/`.

[13] T. Lei and S. Kumar. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In *Euromicro Symposium on Digital Systems Design*, Sept. 1–6 2003.

[14] Mentor Graphics. Inventra intellectual property cores. `http://www.mentor.com/inventra/cores/`.

[15] S. Murali and G. D. Micheli. Bandwidth-constrained mapping of cores onto NoC architectures. In *Design, Automation, and Test in Europe*, pages 896–901. IEEE Computer Society, Feb. 16–20 2004.

[16] Philips Electronics. Philips' IP portfolio. `http://www.semiconductors.philips.com`.

[17] T. Sikora. The MPEG-4 video standard verification model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):19–31, Feb. 1997.

[18] D. Sylvester and K. Keutzer. A global wiring paradigm for deep submicron design. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 19(2):242–252, Feb. 2000.

[19] C. L. Valenzuela and P. Y. Wang. VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions. *IEEE Transactions on Evolutionary Computation*, 6(4):390–401, 2002.

[20] VSI Alliance. On-chip bus attributes specification version 1. `http://www.vsi.org/`, Sept. 2001.

[21] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 4(3):257–271, Nov. 1999.